

# Generation of Development Environments for the Arden Syntax

Magnus Bång, M.Sc., and Henrik Eriksson, Ph.D.

Department of Computer and Information Science, Linköping University

S-581 83 Linköping, Sweden

E-mail: {magba, her}@ida.liu.se

*Providing appropriate development environments for specialized languages requires a significant development and maintenance effort. Specialized environments are therefore expensive when compared to their general-language counterparts. The Arden Syntax for Medical Logic Modules (MLM) is a standardized language for representing medical knowledge. We have used PROTÉGÉ-II, a knowledge-engineering environment, to generate a number of experimental development environments for the Arden Syntax. MÉDAILLE is the resulting MLM editor, which provides a user-friendly environment that allows users to create and modify MLM definitions. Although MÉDAILLE is a generated editor, it has similar functionality, while reducing the programming effort, as compared to other MLM editors developed using traditional programming techniques. We discuss how developers can use PROTÉGÉ-II to generate development environments for other standardized languages and for general programming languages.*

## INTRODUCTION

Software-development tools are computer programs that aid or automate the software-development process. The aim for such tools is to decrease the development effort and to increase the reliability and quality of software. Although there are commercially-available development environments that support this task, these environments are often inflexible in terms of their configuration capabilities for different tasks. Specialized custom-tailored development tools can increase software quality by supporting the projects, the coding and documentation routines, and the target applications. Providing tools, such as development environments for specialized languages, requires a significant development effort. As a result, development environments for such languages are expensive.

Knowledge-acquisition tools (KA-tools) are computer programs that are designed for eliciting knowledge from domain experts and knowledge engineers.

Developers have used meta-tools to generate successful KA-tools for various domains, such as elevator configuration and for the well-known INTERNIST-I system [1,2]. By adopting the meta-tool approach it is possible to provide appropriate development environments for general and specialized programming languages.

An MLM editor is a KA-tool that we can regard as a basic development environment for a specialized language. Several research groups have developed MLM editors using traditional programming approaches. Generally, these implementations are complex and difficult to maintain. To overcome these problems, we have used PROTÉGÉ-II to generate a number of development environments for the Arden Syntax.

The PROTÉGÉ-II [3] approach provides a methodology and a set of programming tools that support the software developer in developing knowledge-based systems using reusable components. Originally, the task of PROTÉGÉ-II was to generate domain-specific KA-tools from domain ontologies. During this work, however, we have established that PROTÉGÉ-II can be used to generate development environments for general programming languages as well. The case study contributes to the understanding of the generality of the meta-tool approach and the principles and benefits of PROTÉGÉ-II.

## BACKGROUND

### The Arden Syntax

When developing medical decision-support systems, a common experience is that the creation of large, medical knowledge bases is a time-consuming and difficult task [4]. Sharing knowledge among departments is therefore desirable. The Arden Syntax [5] for Medical Logic Modules is a standardized language for representing medical knowledge. It is designed for knowledge sharing, and is derived largely from the HELP [6] and RMRS [7] systems.

```

Maintenance:
  title:      Screen for hypokalemia with digoxin therapy;;
  filename:   hypokalemia_and_digoxin;;
  version:    1.06;;
  institution: Columbia-Presbyterian Medical Center;;
  author:     George Hripcsak, M.D.;;
              (hripcs@cumc.columbia.edu);;
  specialist: George Hripcsak, M.D.;;
  date:       1993-09-17;;
  validation: production;;
Library:
  purpose:    Warn the health care provider of hypokalemia
              in the setting of digoxin therapy;;
  explanation: Whenever a serum or whole blood potassium
              value is stored, it is checked for hypokalemia
              (less than 3.3);;
  keywords:   hypokalemia; digoxin; arrhythmia;;
  citations:   1. International Committee of Medical Journals.
              NEJM 1991;324:424-8;;
              CTIM-1.14.5;;
links:
Knowledge:
  type:       data-driven;;
  data:       K:=read last {serum potassium} where it
              occurred before now;;
  priority:   50;;
  evoke:      potassium_storage;;
  logic:      If K>=3.3 then conclude false; endif;;
  action:     write "This patient has hypokalemia in the
              setting of digoxin therapy..."
  urgency:    50;;
End:

```

FIGURE 1. A sample Medical Logic Module<sup>1</sup> that provides alerts regarding a patient's blood status.

The knowledge consists of a set of independent modules, Medical Logic Modules (MLMs). Decision-support systems can use MLMs to provide alerts, management suggestions, data interpretations, treatment schemes, diagnosis scores and so on. An MLM is a text file divided into three categories: *library*, *maintenance* and *knowledge*. The categories consist of a number of slots. The slots of the library and maintenance categories support maintenance of the MLM. Slots of the knowledge category embody the actual knowledge, which for example consists of Pascal-like conditional rules that define medical facts. Figure 1 shows an MLM.

### Development Environments and MLM Editors

The Arden Syntax is similar to traditional procedural programming languages; the MLMs are text files with a well-defined syntax, and the required functionality for MLM editors is similar to the requirements for development environments for programming. In both cases, there is a need for syntax checking, the means to build, debug and run code, and to evaluate statements. Other desirable features are management of code, version handling and generation of dependency graphs. Hence, we can view MLM editors as development environments for a specialized programming language, in this case the Arden Syntax.

Due to the resemblance between the Arden Syntax and many procedural programming languages, we believe that the task of building MLM editors is an appropriate test case for evaluation of the PROTÉGÉ-II approach when generating development environments for general programming languages.

### PROTÉGÉ-II

PROTÉGÉ-II [3] is a methodology and a knowledge-engineering environment that supports developers and domain experts when developing knowledge-based systems. In PROTÉGÉ-II, the programmer can select problem-solving methods from a library of reusable components and configure these to perform tasks needed in the application.

DASH [8] is a meta-level tool within the PROTÉGÉ-II architecture that provides the means to develop KA-tools by automated rapid prototyping. The basis for generation of KA-tools is a declarative ontology specification [9]. DASH takes an ontology as input and automatically generates the KA-tool desired.

PROTÉGÉ-II has been used to develop a wide variety of knowledge-based systems in several application domains, including protocol-based medical care [9], configuration of elevators based on engineering and safety constraints [1], and a reconstruction of INTERNIST-I [2].

### Development of KA-tools

An ontology, in the meta-tool context, denotes a declarative specification that models certain aspects of the real world, for example the relationships among objects in a domain. An ontology can be defined by a hierarchy of classes, where each class represents an object of the domain [10]. Typically, each class has *attributes*, called *slots*, which model the properties of the class.

When generating KA-tools using PROTÉGÉ-II, the first step is to model and characterize the terminology and relations of the problem domain. This step is the ontology development. MODEL [11] is the language for representing ontologies in PROTÉGÉ-II, and is a frame-based knowledge-representation language based on COOL, the object-oriented part of the CLIPS language [12]. The second step is to run DASH on the ontology and establish the layout design. Here, the developer can resize and reposition widgets, and change labels by direct manipulation using an interface builder. The output from DASH is a declarative specification of the target tool which can be executed either by a run-time system or be compiled into C code [11].

<sup>1</sup> Taken from the public library of MLMs at the Columbia-Presbyterian Medical Center.

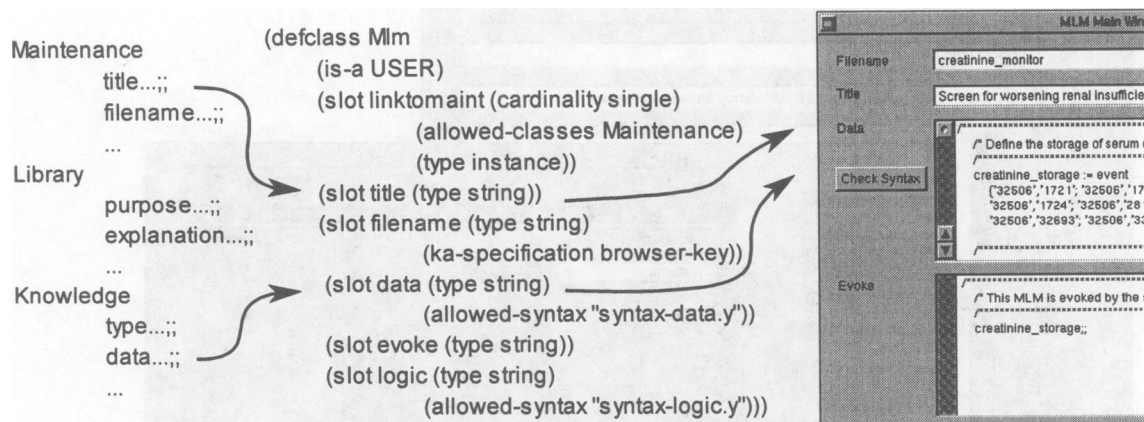


FIGURE 2. The mapping among the MLMs, ontology and form of the resulting development environment. The “title” slot (with the specified type “string”) in the ontology represents the title slot of the MLM. The ontology slot has its corresponding widget in the form generated by DASH.

## ONTOLOGY DEVELOPMENT

We now discuss issues concerning how we can develop an ontology for the Arden Syntax. By studying the Arden Syntax definition (in BNF) and the semantics of the MLM slots, we can establish the mapping among the slots defined in the Arden Syntax specification and the classes and slots of the ontology. One approach is to create an “MLM-class” that consists of *all* the slots of the MLM. However, we abandoned this solution due to reasons concerning the human-computer interface. In this case, DASH would generate a single form with all the slots of the MLM. When working with editors in daily practice, users generally find single forms undesirable because of information overload<sup>2</sup>. The majority of the slots support maintenance and hence the editor should not display them by default. We have translated the MLM to a number of classes, representing different parts of the MLM. Figure 2 illustrates how we model part of the MLM in the ontology.

We developed incrementally three different ontologies and therefore three different versions of the MLM editor using DASH. First, we developed a prototype that was a replica of an existing system developed at the Department of Biomedical Engineering, Medical Informatics at Linköping University. This established the validity of the PROTÉGÉ-II approach in terms of generating basic editors for MLMs. Second, by reusing and modifying the previous ontology, we developed a new version with increased functionality such as organization of the MLMs. MÉDAILLE is the final version of the MLM editor. We developed it in the same manner by reusing and modifying the ontology from the previous version.

## MÉDAILLE

MÉDAILLE is a generated development environment for the Arden Syntax that manages a database of MLMs and provides support for entering and editing the MLMs. Figure 3 shows part of the user interface of MÉDAILLE. The organization of the MLMs is a hierarchical structure with respect to the medical domain and author. Syntax checking of slots and integrated terminology support in terms of a semantic network of interrelated frames are also available. MÉDAILLE has on-line help that provides support for Arden Syntax and how to use the editor.

Let us compare MÉDAILLE and two editors developed at the Department of Biomedical Engineering, Medical Informatics at Linköping University [13]. Table 1 provides a functional comparison. The first editor, SEMLA [14], runs under the X Window System. The development tool used to produce SEMLA was TeleUSE [15] together with a toolbox for compiler construction. The second editor runs under the MS Windows environment [16]. The basis for this editor is SEMLA, and the development tool was Visual C++. The comparative study shows that the MLM editors generated by PROTÉGÉ-II embody much of the behavior provided by editors developed using traditional programming approaches.

TABLE 1. Comparison of functionality in the PROTÉGÉ-II, TeleUSE and Visual C++ approaches.

Available Functions	PROTÉGÉ	TeleUSE	Visual C++
Basic Editing	✓	✓	✓
Syntax Checking	✓	✓	✓
Terminology	✓		
MLM Organization	✓		✓
Version Control			
On-line Help	✓	✓	✓
Syntax-directed Editing		✓	

<sup>2</sup> Personal communication, Dr. N. Shahsavar, Linköping University.

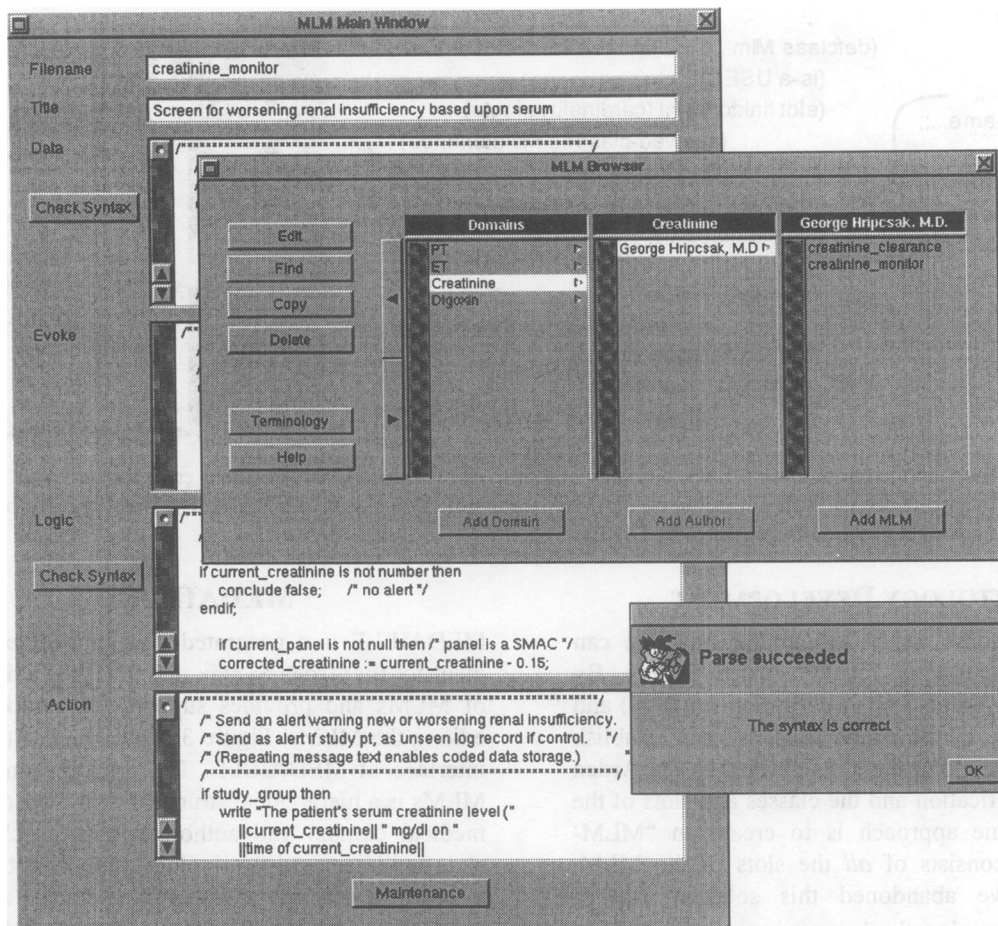


FIGURE 3. A part of MÉDAILLE, the MLM editor generated by PROTÉGÉ-II.

Using PROTÉGÉ-II, we produced three different versions of the MLM editor in just 14 person-weeks. Using Visual C++, the development effort for an editor was 15 person-weeks and 30 person-weeks for the SEMLA editor.

## DISCUSSION

As we have seen, it is possible take the PROTÉGÉ-II approach beyond the generation of traditional KA-tools. Because the architecture supports generation of development environments for the Arden Syntax, it should be possible to use the same approach for other programming languages, such as C, Java and Ada. The advantages that PROTÉGÉ-II provides when building traditional KA-tools, thus extend to the generation of development environments for general programming languages.

We found that the main advantage of applying the meta-tool approach is the flexibility that it provides. This flexibility is manifold. Rapid prototyping provides a means to reconsider the target tool if it is inadequate. A radical change in the fundamental definitions of the target tool is unproblematic. It is a relatively easy task to regenerate it with some minor changes in the domain ontology and in the BNF grammar files. In addition, it is relatively easy to port the target tool to other

platforms. We ported the MLM ontology to the MS Windows version of PROTÉGÉ-II, Protégé/Win. As part of this process, we made minor adjustments in the ontology and in the graphical user interface. The effort required for the port was about one person-week. Furthermore, it is possible to reuse parts of the target tools (perhaps developed by others) and integrate them into a new tool by merging the ontologies.

PROTÉGÉ-II supports the developer in custom-tailoring traditional KA-tools to suit the domain experts' needs and the problem domain. In the case of development environments for programming languages, the environments can be custom-tailored to suit the users and the situation, in this case programmers, programming languages and target applications. Thus, it is possible to create development tools that are in between general development environments for programming languages and domain-oriented KA-tools. Such tools could increase software quality by supporting project-specific programming and maintenance (e.g., by supporting the local coding and documentation routines, and the target applications).

Although PROTÉGÉ-II supports layout-oriented custom adjustments of user interfaces, we found that there are other important features that should be

configurable and extendible. The current version of PROTÉGÉ-II does not provide any Application Programming Interface (API) or scripting language that developers can use to further custom-tailor the target tools (e.g., to provide syntax-directed editing support for expressions). Other desirable, but currently not supported, features for generating development environments are version control, debugger tool integration, code builder, graphical tree browsers and hypertext support. Although it is possible to extend PROTÉGÉ-II with some of these features, there are still advantages in providing an API. For example, the benefit of the architecture would increase significantly if developers could implement and share their own extensions. Moreover, the meta-tool architecture itself could consist of a set of plug-in components.

### CONCLUSION

We approached the problem of implementing development environments for the Arden Syntax by using PROTÉGÉ-II. The resulting MLM editor, MÉDAILLE, is similar to other hand-crafted MLM editors. We found that the approach decreases the maintenance problems, reduces the development effort and provides a means to custom-tailor the development environment. The work illustrates that PROTÉGÉ-II can be used to generate not only traditional KA-tools, but also development environments for general and specialized programming languages. To support full development of such environments, PROTÉGÉ-II needs an API as well as programming-support components (e.g., compiler and debugger interfaces). Nevertheless, we believe that PROTÉGÉ-II and its MS Windows version, Protégé/Win, are appropriate platforms for the design of such meta-level environments.

### Acknowledgments

We would like to thank Dr. Nosrat Shahsavari and Professor Toomas Timpka for their comments and suggestions.

The work has been supported in part by the Swedish National Board for Industrial and Technical Development (NUTEK), grant no. 93-3233 and the Swedish Research Council for Engineering Sciences (TFR), grant no. 95-186.

The PROTÉGÉ-II research project URL:  
<http://smi.stanford.edu/projects/protege>.

The MÉDAILLE homepage URL:  
<http://www.ida.liu.se/~magba/medaille.html>.

### References

1. Gennari JH, Altman RB and Musen MA. Reuse with PROTÉGÉ-II: From elevators to ribosomes. *Proceedings of the ACM-SIGSOFT Symposium on Software Reusability*. Seattle, WA. 1995:72-80.
2. Musen MA, Gennari JH, & Wong WW. A Rational Reconstruction of INTERNIST-I using PROTEGE-II. *19th Annual Symposium on Computer Applications in Medical Care*. New Orleans, L.A. 1995:289-293.
3. Musen MA, Gennari JH, Eriksson H, Tu SW, Puerta, AR. PROTÉGÉ-II: Computer support for development of intelligent systems from libraries of components. *Proceedings of Medinfo'95*. Vancouver, BC; 1995:766-770.
4. Gao X, Realizing Medical Decision Support using the Arden Syntax as Knowledge Representation. Licentiate Thesis no. 399. Linköping Studies in Science and Technology. 1993.
5. Hripcsak G, Clayton PD, Pryor TA, Haug P, Wigertz OB, Van der lei J. The Arden Syntax for Medical Logic Modules. *Proceedings of the 14th Annual Symposium on Computer Applications in Medical Care*. Washington, D.C. 1990:200-204.
6. HELP Frame Manual. Version 1.6. Salt Lake City:LDS Hospital. 1989.
7. McDonald CJ, Action-Oriented Decisions in Ambulatory Medicine. Chicago: Year Book Medical Publishers. 1981.
8. Eriksson H, Puerta AR, Musen MA. Generation of knowledge acquisition tools from domain ontologies, *International Journal of Human Computer Studies*. 1994;41:425-453.
9. Musen MA, Tu SW, Das AK, Shahar Y. EON: A Component-Based Approach To Automation of Protocol-Directed Therapy. *Journal of the Medical Informatics Association*. 1996;3(6): 367-388.
10. Neches R, Fikes R. Enabling technology for knowledge sharing. *AI Magazine*. 1991;12(3): 36-58.
11. Gennari JH. A brief guide to MAITRE and MODEL: an ontology editor and a frame-based knowledge representation language. Working Paper KSL 93-45. Stanford University, Palo Alto, CA. 1993.
12. NASA. CLIPS Reference Manual. Software Technology Branch. Lyndon B. Johnson Space Center. Houston, TX. 1991.
13. Bång M. Automated Generation of Editors for Medical Logic Modules Using Ontologies. M.Sc. Thesis LiTH-IDA-Ex-9650. Linköping University. 1996.
14. Carlsson M, Ohlsson P. SEMLA - ett editerings-verktyg för medicinsk kunskapsinsamling enligt Arden syntax [SEMLA - An Editor for Medical Knowledge-Acquisition for the Arden Syntax], M.Sc. Thesis ULi-IMT-EX-191, Linköping University, 1991.
15. Alsys Inc. TeleUSE System Overview. Alsys Inc San Diego, CA, 1994.
16. Rosengren O. Utveckling av kunskapsbas-editor för Arden syntax [Development of a Knowledge-Base Manager Using Arden Syntax], M.Sc. Thesis LIU-IMT;143, Linköping University. 1995.